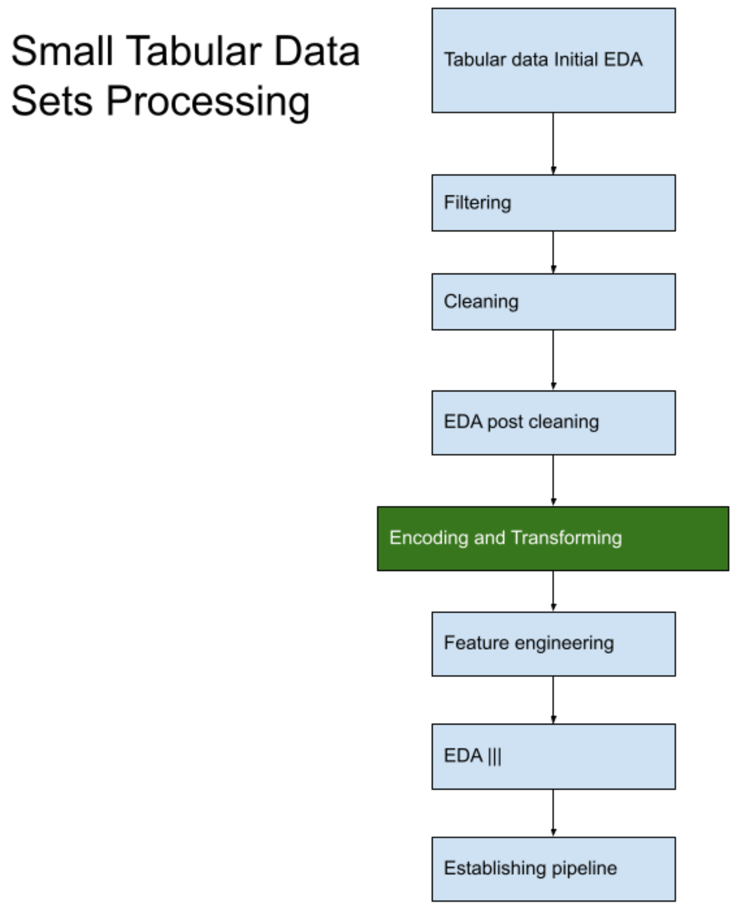
Encoding and Transforming



1. Encoding:
   1.1. Categorical variables :
      1.1.1. One-Hot Encoding: This method creates a binary column for each category level. It's widely used but can lead to a high dimensionality if the categorical feature has many levels.
         1.1.1.1. Use Case: Suitable for categorical variables with a relatively low number of unique values. Commonly used in linear models, decision trees, and machine learning algorithms that require numerical input and do not handle categorical data intrinsically.
         1.1.1.2. Example: Encoding a 'Color' feature with categories like 'Red', 'Blue', 'Green' into separate binary columns.

1.1.2. Label Encoding: Each category is assigned a unique integer based on alphabetical ordering. This method is simple but implies an ordinal relationship that might not exist, which can lead to poor performance or misleading results in some models.

   1.1.2.1. Use Case: Best for ordinal categorical variables where the order of categories is meaningful, such as 'Low', 'Medium', 'High'. It's also useful when memory efficiency is crucial, as it creates a single new feature.

   1.1.2.2. Example: Encoding 'Size' with values 'Small', 'Medium', 'Large' into 1, 2, 3.

1.1.3. Ordinal Encoding: Similar to label encoding, but the integers assigned to categories are based on their order or rank. This is suitable for ordinal data where the order of categories is meaningful.

   1.1.3.1. Use Case: Similar to label encoding but specifically when the categorical variable has an inherent order and the order is explicitly known.

   1.1.3.2. Example: Encoding 'Education Level' with values 'High School', 'Bachelor', 'Masters', 'PhD' in increasing order of education level.

1.1.4. Binary Encoding: This method combines the features of both one-hot and label encoding. Categories are first converted to numeric labels and then those numbers are converted into binary numbers. Each binary digit gets one column.

   1.1.4.1. Use Case: Useful for high cardinality features. It is more efficient than one-hot encoding, as it requires fewer new columns.

   1.1.4.2. Example: Encoding a 'City' feature with numerous categories into binary digits.

1.1.5. Frequency Encoding: Categories are replaced by the frequency or percentage of the time they occur in the dataset. This can be useful when the frequency of categories is predictive.

- 1.1.5.1. Use Case: Effective for categorical variables with many categories. Helps algorithms understand the frequency of occurrence, which could be an important factor.
- 1.1.5.2. Example: Encoding 'Product Type' in a sales dataset by the frequency of each product type.

1.1.6. Mean Encoding (or Target Encoding): Categories are replaced with the mean of the target variable for that category. This is particularly useful for high-cardinality categorical features but can lead to overfitting.

- 1.1.6.1. Use Case: Particularly useful in supervised learning where the target variable's mean for each category can be a strong indicator. However, it risks overfitting and needs careful validation.
- 1.1.6.2. Example: In predicting credit default, encoding 'Occupation' with the average default rate for each occupation.

1.1.7. Hashing: The categories are hashed to a fixed size of columns. This can be useful when dealing with a large number of categories, but information can be lost due to collisions in the hashing process.

- 1.1.7.1. Use Case: Beneficial for very large datasets or when the number of categories is huge or can grow over time.
- 1.1.7.2. Example: Encoding user-generated tags in an online platform, where new tags might be added regularly.

1.1.8. Backward Difference Encoding: This method compares the mean of the dependent variable for a level to the mean of the dependent variable for the prior level. This is particularly useful in linear regression models.

- 1.1.8.1. Use Case: Beneficial for very large datasets or when the number of categories is huge or can grow over time.
- 1.1.8.2. Example: Encoding user-generated tags in an online platform, where new tags might be added regularly.

1.1.9. Helmert Encoding: It compares each level of a categorical variable to the mean of the subsequent levels.

1.1.9.1. Use Case: Suitable for ordinal variables in linear regression models. It compares the mean of the dependent variable for each level to the mean of the subsequent levels.

1.1.9.2. Example: Encoding 'Service Tiers' like Bronze, Silver, Gold, and Platinum, where each is compared to the mean of higher tiers.

1.1.10. BaseN Encoding: A generalized version of binary encoding, where categories are first converted to numerical labels and then those numbers are converted into Base-N numbers.

1.1.10.1. Use Case: A flexible method that's useful for a trade-off between one-hot and binary encoding. It's advantageous when the number of categories is moderately high.

1.1.10.2. Example: Encoding 'Country' in a global dataset where BaseN allows controlling the number of new columns.

2. Numerical Variables

2.1. Normalization (Min-Max Scaling): This transformation rescales the data to a fixed range, usually 0 to 1. It's useful when you need to compare data that's measured on different scales.

2.2. Difference with respect to aggregation i.e. the value difference from the mean, median etc.

2.3. Standardization (Z-score Normalization): This involves rescaling the data so it has a mean of 0 and a standard deviation of 1. It's useful in algorithms that assume the data is normally distributed.

2.4. Log Transformation: Useful for dealing with skewed data, log transformation can help stabilize the variance and make the data more 'normal' for certain types of analysis.

2.5. Square Root Transformation: This is another method to reduce right skewness, similar to log transformation but not as strong.

2.6. Box-Cox Transformation: A more generalized form of power transformation that can handle positive values, useful for stabilizing variance and making the data more normal.

2.7. Yeo-Johnson Transformation: Similar to Box-Cox but can handle both positive and negative values.

2.8. Exponential and Power Transformations: These are useful for increasing the spread of data that is highly concentrated.

2.9. Binning or Quantization: This involves converting numerical data into categorical bins, which can be useful for non-parametric approaches or to handle outliers.

2.10. Winsorizing: This involves capping the extreme values at a certain percentile to reduce the effect of outliers.

2.11. Rank Transformation: Converts the data into their rank order. This is useful when the data is not normally distributed and you're more interested in the order rather than the actual values.

2.12. Inverse Transformation: This is particularly useful for data following an inverse relationship. It can be effective in handling certain types of skewness.

2.13. Square Transformation: Applied to positively skewed data, the square transformation can sometimes normalize data.

2.14. Cube Root Transformation: This transformation is less severe than a log transformation and can be applied to both positive and negative values. It's good for reducing skewness in data.

2.15. Hyperbolic Sine Transformation (Sinh): This is a less common transformation but can be useful for certain distributions, especially when both positive and negative values need to be transformed symmetrically.

2.16. Logit Transformation: Used primarily for data that is bounded between 0 and 1, such as proportions or percentages. It's a good choice for transforming data that follows a binomial distribution.

2.17. Arcsin Transformation: This is another transformation for data that lies between 0 and 1 (like proportions and percentages), and it can help stabilize the variance in this range.

2.18. Fisher’s Z Transformation: This is often used for transforming correlation coefficients to stabilize the variance.

2.19. Discrete Fourier Transform (DFT): For time-series data, DFT can be used to transform the data into frequency domain, which is particularly useful in identifying cyclic patterns.

2.20. Generalized Log Transformation: This is a variation of the log transformation that can be applied to zero or negative values as well as positive values.

2.21. Beta Transformation: This is useful for variables that are bounded on both sides (like proportions and percentages) and follow a beta distribution.

2.22. Quantile Transformation: This transformation maps the data to a desired distribution, like the normal or uniform distribution, based on quantiles. This is useful when you want your data to adhere to a specific distribution shape.